Restaurant Delivery Time Prediction using Python

Analysis & Prediction

Course : Machine Learning (CT-562)

Project Advisor : Dr. Haider Ali

Presenter : Presenter : Danish Akhund

Table of Contents

Background of Project

1. The Machine learning is among the miracle for solving complex problems. For selection of appropriate and applicable projects we take help from internet source topic names ‘290+ Machine Learning Projects with Python’.
2. From reviewing these bundle of projects we converge on two projects one is News classification related to Data cleaning and the other is “Food Delivery Time Prediction”.
3. By discussing with course advisor we come with an idea to work on Food delivery project.
4. We than review , research on the similar projects done in the past to search room of further enhancement in this project. The notable

Objective of Project ? Why use Restaurant Delivery Time case study

Practical life example for machine Learning

These analysis of prediction models will help to save fuel.

Dataset Acquisition

Food Delivery Time Prediction: Case Study

JANUARY 2, 2023

Download the dataset below to solve this Data Science case study on food delivery time prediction. (Dataset Source: **Kaggle**)

Download Data

Food Delivery Time Prediction: Case Study

Predicting the delivery time of your order is a challenging task for every food delivery service like Zomato and Swiggy.

One of the best strategies to predict the delivery time is by calculating the distance between the point of picking up the order and the point of delivering the order. And then predicting the delivery time based on how much time your delivery partners took to deliver orders in the past for the same distance.

Data set Acquisition = Continue

1. ID: order ID number
2. Delivery_person_ID: ID number of the delivery partner
3. Delivery_person_Age: Age of the delivery partner
4. Delivery_person_Ratings: ratings of the delivery partner based on past deliveries
5. Restaurant_latitude: The latitude of the restaurant
6. Restaurant_longitude: The longitude of the restaurant
7. Delivery_location_latitude: The latitude of the delivery location
8. Delivery_location_longitude: The longitude of the delivery location
9. Type_of_order: The type of meal ordered by the customer
10. Type_of_vehicle: The type of vehicle delivery partner rides
11. Time_taken(min): The time taken by the delivery partner to complete the order

You are required to predict the delivery time based on the distance covered by the delivery partner to deliver the order.

Importing libraries

```
import pandas as pd
import numpy as np
import plotly.express as px

data = pd.read_csv("deliverytime.txt")
print(data.head())
```

01.Function /code for Pandas read.
02. The head() function is used to get the first few rows.

```
     ID Delivery_person_ID  Delivery_person_Age  Delivery_person_Ratings  \
0  4607     INDORES13DEL02                   37                      4.9
1  B379     BANGRES18DEL02                   34                      4.5
2  5D6D     BANGRES19DEL01                   23                      4.4
3  7A6A    COIMBRES13DEL02                   38                      4.7
4  70A2     CHENRES12DEL01                   32                      4.6

   Restaurant_latitude  Restaurant_longitude  Delivery_location_latitude  \
0            22.745049             75.892471                   22.765049
1            12.913041             77.683237                   13.043041
2            12.914264             77.678400                   12.924264
3            11.003669             76.976494                   11.053669
4            12.972793             80.249982                   13.012793

   Delivery_location_longitude Type_of_order Type_of_vehicle  Time_taken(min)
0                    75.912471        Snack      motorcycle               24
1                    77.813237        Snack         scooter               33
2                    77.688400       Drinks      motorcycle               26
3                    77.026494       Buffet      motorcycle               21
4                    80.289982        Snack         scooter               30
```

Overview of the Data

```
In [2]: # columns information
        data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45593 entries, 0 to 45592
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   ID                           45593 non-null  object
 1   Delivery_person_ID           45593 non-null  object
 2   Delivery_person_Age          45593 non-null  int64
 3   Delivery_person_Ratings      45593 non-null  float64
 4   Restaurant_latitude          45593 non-null  float64
 5   Restaurant_longitude         45593 non-null  float64
 6   Delivery_location_latitude   45593 non-null  float64
 7   Delivery_location_longitude  45593 non-null  float64
 8   Type_of_order                45593 non-null  object
 9   Type_of_vehicle              45593 non-null  object
 10  Time_taken(min)              45593 non-null  int64
dtypes: float64(5), int64(2), object(4)
memory usage: 3.8+ MB
```

This method prints information about a DataFrame including the index dtype and columns, non-null values and memory usage. Parameters. verbosebool, optional.

45 593 entries

11 columns

Looking for null

In [3]:
```
# finding Null values
data.isnull().sum()
```

Out[3]:
```
ID                             0
Delivery_person_ID             0
Delivery_person_Age            0
Delivery_person_Ratings        0
Restaurant_latitude            0
Restaurant_longitude           0
Delivery_location_latitude     0
Delivery_location_longitude    0
Type_of_order                  0
Type_of_vehicle                0
Time_taken(min)                0
dtype: int64
```

Calculating Distance Between Two Latitudes and Longitudes

```
R = 6371

# Convert degrees to radians
def deg_to_rad(degrees):
    return degrees * (np.pi/180)

# Function to calculate the distance between two points using the haversine formula
def distcalculate(lat1, lon1, lat2, lon2):
    d_lat = deg_to_rad(lat2-lat1)
    d_lon = deg_to_rad(lon2-lon1)
    a = np.sin(d_lat/2)**2 + np.cos(deg_to_rad(lat1)) * np.cos(deg_to_rad(lat2)) * np.sin(d_lon/2)**2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))
    return R * c
```

haversine(θ) = $\sin^2(\theta/2)$. The haversine formula is a very accurate way of computing distances between two points on the surface of a sphere using the latitude and longitude of the two points

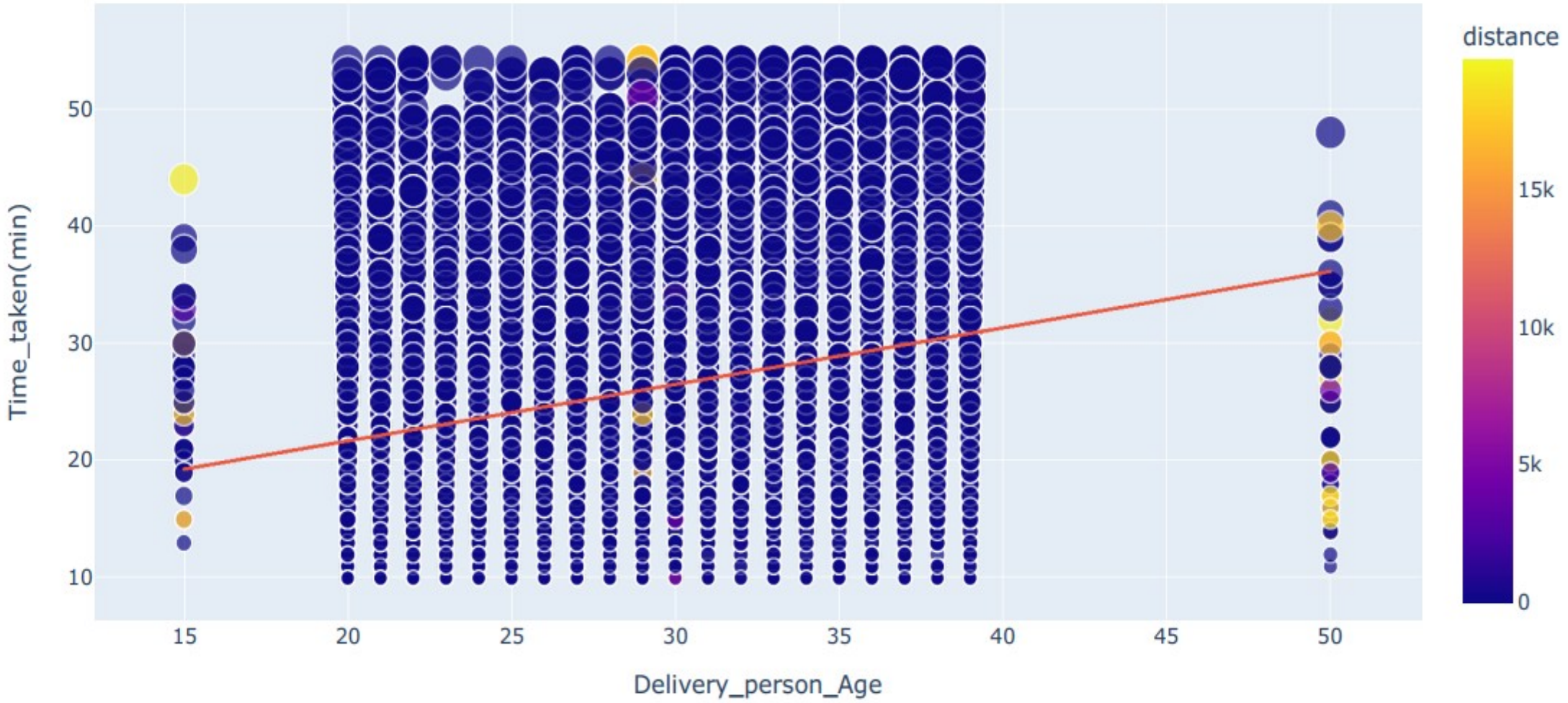
Exploring data Analysis

Relationships b/w Features

First Relationships b/w Time & Age of rider

```
figure = px.scatter(data_frame = data,
                    x="Delivery_person_Age",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color = "distance",
                    trendline="ols",
                    title = "Relationship Between Time Taken and Age")
figure.show()
```

Relationship b/w Time Taken & Age

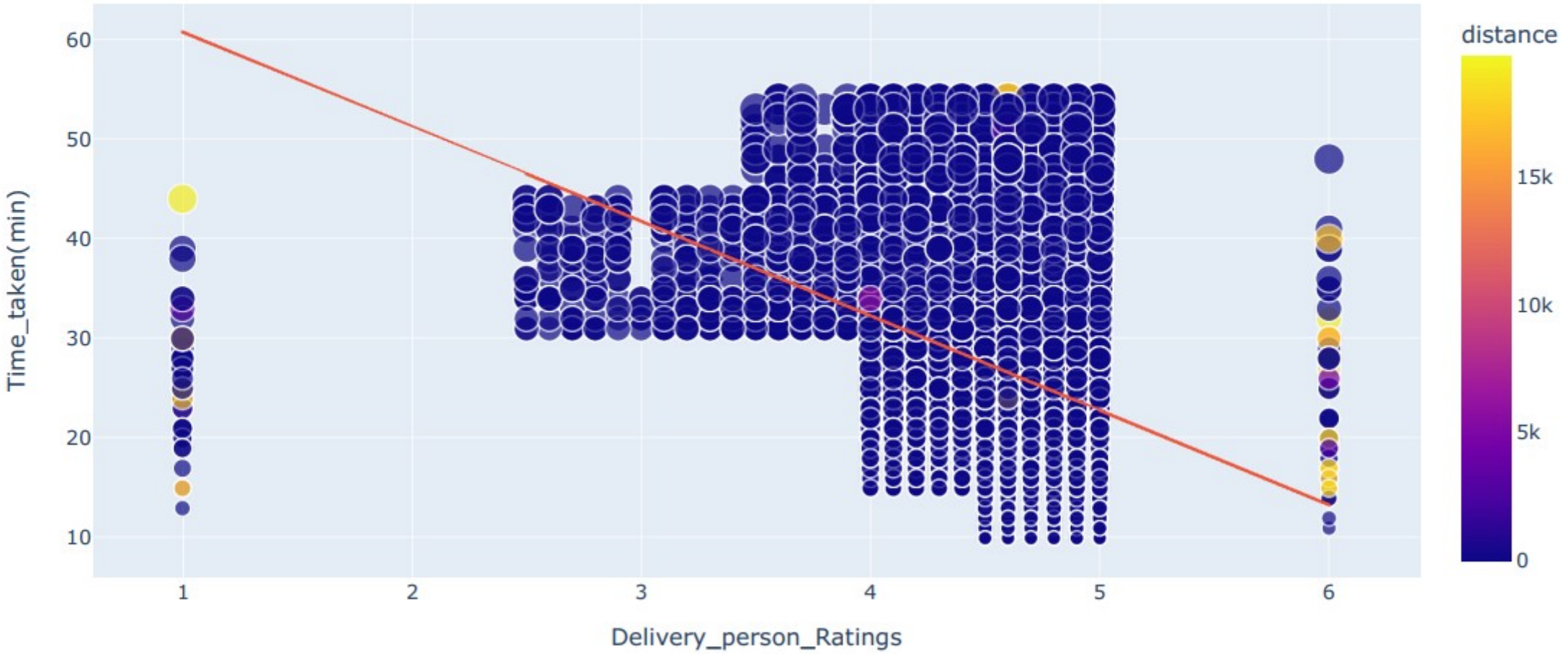


Relationships b/w Features

econd Relationship Between Time Taken and Ratings

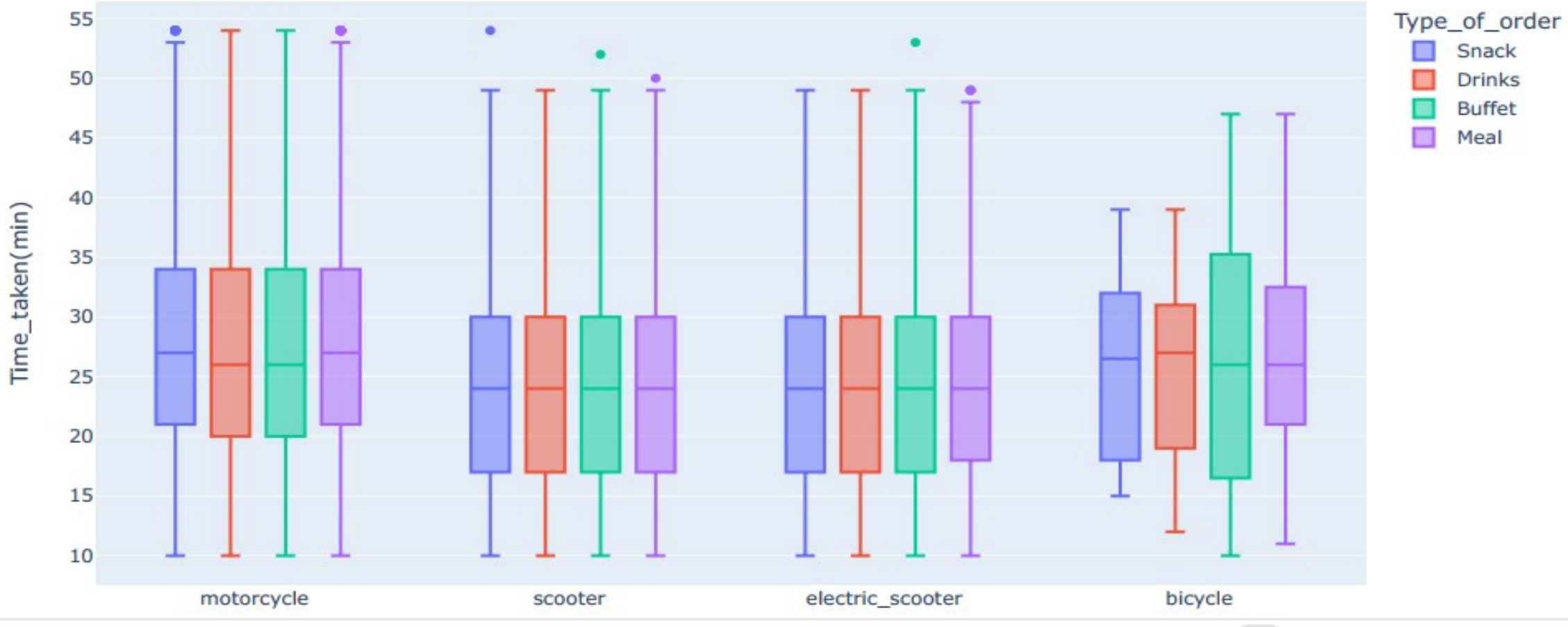
```
figure = px.scatter(data_frame = data,
                    x="Delivery_person_Ratings",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color = "distance",
                    trendline="ols",
                    title = "Relationship Between Time Taken and Ratings")
figure.show()
```

Relationship Between Time Taken and Ratings



Relation b/w Type of Order , Vehicle & Time

```
fig = px.box(data,
             x="Type_of_vehicle",
             y="Time_taken(min)",
             color="Type_of_order")
fig.show()
```



Insight

So , there is very less difference between the time taken by delivery partners depending on the vehicle.

Hence, Contributing Features:

1. Delivery partners' Age.
2. Delivery Partner Ratings.
3. Distance b/w the restaurant & delivery location.

Delivery Time Prediction

```
from sklearn.model_selection import train_test_split
x = np.array(data[["Delivery_person_Age","Delivery_person_Ratings","distance"]])
y = np.array(data[["Time_taken(min)"]])
xtrain, xtest, ytrain, ytest = train_test_split(x, y,test_size=0.10,random_state=42)
```

```
from keras.models import Sequential
from keras.layers import Dense, LSTM
model = Sequential()
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (xtrain.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 3, 128)            66560

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 25)                1625

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 117619 (459.45 KB)
Trainable params: 117619 (459.45 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Training the model

```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(xtrain, ytrain, batch_size=1, epochs=9)
```

```
Epoch 1/9
41033/41033 [==============================] - 365s 8ms/step - loss: 69.4178
Epoch 2/9
41033/41033 [==============================] - 357s 9ms/step - loss: 63.4860
Epoch 3/9
41033/41033 [==============================] - 355s 9ms/step - loss: 61.3758
Epoch 4/9
41033/41033 [==============================] - 355s 9ms/step - loss: 60.6749
Epoch 5/9
41033/41033 [==============================] - 363s 9ms/step - loss: 59.6832
Epoch 6/9
41033/41033 [==============================] - 360s 9ms/step - loss: 59.7502
Epoch 7/9
41033/41033 [==============================] - 384s 9ms/step - loss: 59.8624
Epoch 8/9
41033/41033 [==============================] - 387s 9ms/step - loss: 59.2573
Epoch 9/9
36427/41033 [=========================>....] - ETA: 41s - loss: 59.4699
```

Prediction on Our Model

```
print("Food Delivery Time Prediction")
a = int(input("Age of Delivery Partner: "))
b = float(input("Ratings of Previous Deliveries: "))
c = int(input("Total Distance: "))

features = np.array([[a, b, c]])
print("Predicted Delivery Time in Minutes = ", model.predict(features))
```

```
Age of Delivery Partner: 26
Ratings of Previous Deliveries: 5
Total Distance: 25
1/1 [==============================] - 91s 91s/step
Predicted Delivery Time in Minutes =  [[25.485943]]
```

Conclusion

For prediction of time

1. Distance b/w the restaurant and the delivery locations
2. Also to find relationships b/w the time taken by delivery partners to deliver the food in the past for same distance.

References

1. “290+ Machine Learning Projects with Python | by Aman Kharwal | Coders Camp | Medium.”
2. “Food_Delivery_Time_Analysis_Project/MiniProject_FoodDelivery_Final.Ipynb at 08a0ac7a21c47e5f6f967b0453d0037e41e3e9c5 · Dev26git/Food_Delivery_Time_Analysis_Project.”
3. Kharwal, “Food Delivery Time Prediction Using Python | Aman Kharwal.”